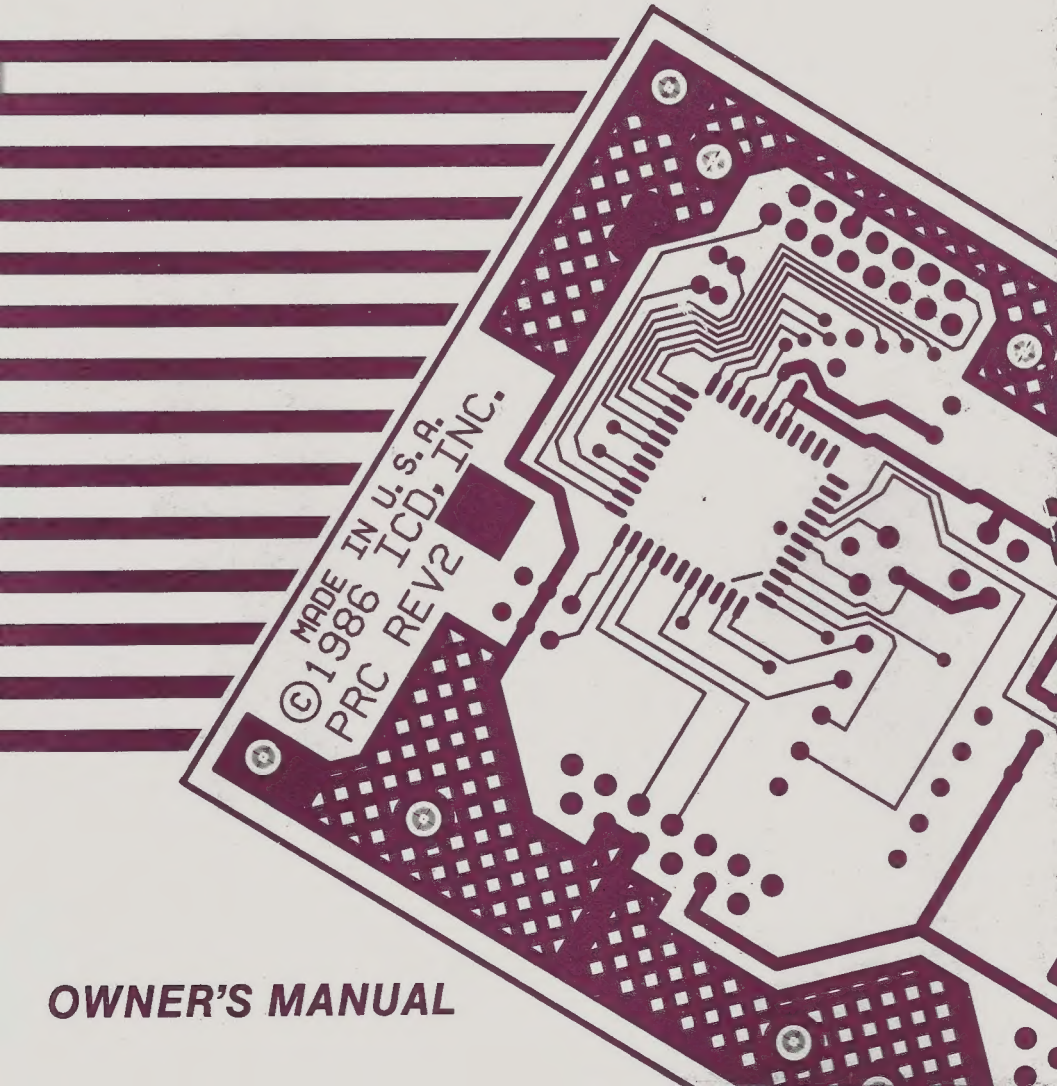
P: R: CONNECTION

The Printer and MODEM Interface for Atari Computers



OWNER'S MANUAL

P:R: Connection

The Printer and
MODEM Interface
for Atari Computers

by ICD

Note—throughout this manual:

SpartaDOS, SpartaDOS Construction Set, UltraSpeed, US Doubler, R-Time 8, RAMBO XL, and P:R: Connection, are trademarks of ICD, Inc.

Atari 850, 130XE, 800XL, and 1200XL, are trademarks of Atari Corp.

Published by ICD, Inc.
1220 Rock Street
Rockford, IL 61101-1437
U.S.A.

PREFACE

You have just purchased the P:R: Connection, another high quality product from ICD. The P:R: Connection has been designed to add lasting value to your 8-bit Atari Computer by allowing you the choice of hundreds of printers (P: devices) and MODEMs (R: devices). Thousands of dollars and many man hours have been used to develop *the most* economical and flexible high quality interface for your needs. There is no such thing as 100% compatibility (as we tried with the 850) but we have come very close. It is impossible to match code byte for byte without using exactly the same hardware (a feat which was not economically feasible). Instead, we created something much better than the 850 for a lower price (much like Atari did when they created the 800XL to replace the old 800 computer). Virtually all printer software (designed for the 850) will work with the P:R: Connection, and most MODEM software will work without any modification. We have included a translator type file (PRC.SYS) which should work with the few MODEM programs which otherwise will not run. (See Appendix F.) For the latest information on P:R: Connection compatibility call the ICD BBS. It's on-line 24 hours a day at 815/968-2229 running 300/1200/2400 baud.

TABLE OF CONTENTS

APPENDICES

WARRANTY

CHAPTER 1—INTRODUCTION TO THE P:R: CONNECTION

Why an interface?

The P:R: Connection is an interface between your 8-bit Atari computer and other RS-232 or 'centronics' parallel devices. These devices may include MODEMs, printers, other computers, or anything which uses either of these two types of ports. There are dedicated MODEMs and printers available just for the Atari which require no interface. These dedicated devices are fine as long as you are satisfied with their operation and never plan on buying another computer. On the other hand, standard serial and parallel devices will work with most other computers including the new Atari ST and the IBM PC.

P: and R: Devices

There have been thousands of programs written for the 8-bit Atari computers many of which use a printer or MODEM ('P:' or 'R:' device). Although some also support other standards, these programs almost always support the Atari 850 interface standard. Before the P:R: Connection, there was no way to accomplish this device standard other than by using an Atari 850. Most of these programs require a 'P:' device for the printer and an 'R:' device for the MODEM. If you do plan on using a serial printer with your Atari, make sure the programs you use will support an 'R:' device for a printer. NOTE: ICD has written a DOS command (SPRINT) which will divert the output and make the 'R:' device look like the 'P:' device to the system. This allows the use of a serial printer (with software requiring a 'P:' device) with any program running under SpartaDOS.

How does it work?

Inside the P:R: Connection is our custom computer chip (PRC9985-6) which contains ROM, RAM, a CPU, and a PIA. This is effectively an entire computer on a single chip. The ROM portion contains the software to make the printer port work like a 'centronics' standard port or 'P:' device. The 'P:' device is virtually identical to the Atari 850 'P:' device.

This ROM also contains the software handler which loads into the Atari computer (when called) and sets up the two serial ports as 'R:' devices ('R1:' and 'R2:'). This 'R:' handler is loaded either with the AUTORUN.SYS (RS232.SYS) which comes with Atari DOS (RS232.COM with SpartaDOS) or else whenever the computer is powered up with the P:R: Connection attached and no disk drives respond. The 'R:' handler is relocatable which means that it loads into the computer at the lowest possible memory location and then protects itself by moving MEMLO up. The P:R: Connection's 'R:' handler is very similar to the Atari 850 'R:' handler at the CIO level. (This means that it uses the same XIO commands as the Atari 850 interface device.) At the SIO level there are several calls different from the 850 which may make a few programs designed for the 850 not function properly. To remedy this, we have included an SIO emulation handler called PRC.SYS. More information on compatibility and the SIO differences can be found in appendices B, C, and F.

Compatibility

The P:R: Connection has been designed as a cost effective replacement for the now obsolete Atari 850 interface. Our hardware design requirements were to make a unit small and compact with clean ergonomic design. This required a molded case with cords only to be attached on two sides. The bulky external power supply also had to be eliminated. All of this was made possible due to the recent development of low power single chip microcomputers. Unfortunately, the 1200XL requires an internal modification to work with the P:R: Connection or any other device which uses the computer for its power. (See appendix E for details.)

The P:R: Connection software design requirements were full compatibility with the 850 protocol. Like the problem Atari had when they designed the 800XL, we found programs written for the 850 which used illegal calls outside the CIO architecture. To provide a link between these programs and the P:R: Connection, we have included a binary file called PRC.SYS which works like a translator. PRC.SYS is fully relocatable and works with any DOS. This should provide full compatibility with programs designed for the 850 which use an 'RBIN' type handler and normally don't load the 850 handler (like Hometerm). If programs are found which don't work properly with the P:R: Connection, we will make every attempt to provide a patch or solution for proper operation. (See appendix F for more information on compatibility.)

Installation and Use

Since the P:R: Connection is powered by the host computer, it should be plugged directly into the 13 pin male socket where the disk drive normally goes. Then plug the next device into the 13 pin socket on the P:R: Connection, the next device into that, and so on. (It will probably work from any of the 13 pin connectors in the daisy chain but since there is a voltage drop in each of these connectors, it is best to plug it directly into the computer.) You will then need a cable to connect between the P:R: Connection ports and the peripheral which you intend to use. These cables may be purchased direct from ICD, from your dealer, or made from the specifications in table 1-1 or 1-2, appendix D, and your peripheral manual. (Cables designed for the Atari 850 will work.) The P:R: Connection ports and their locations are:

R1: This is the 9 pin connector towards the outside. Use this connector as your main RS232 serial port since it supports full handshaking.

R2: This is the nine pin connector in the center. Use this only when you need an extra RS232 port and with softward which supports an 'R2:' device.

P1: 'P1:' is the parallel printer port which is the 15 pin connector located next to 'R2:'.

Using the P:R: Connection with a MODEM

Connect your MODEM cable between serial port 1 (the 9 pin socket on the outside) of the P:R: Connection and your MODEM. For correct operation with a particular program, see your terminal program for details on use with the 850 interface.

TERMINAL PROGRAMS

AMODEM7

Through a special arrangement with Trent Dudley, author of AMODEM7, we have included a full version of his latest terminal program. We feel this is one of the best terminal programs around for Atari computers. AMODEM7 is a BASIC program with machine code speed. It works at 300, 1200, 2400 baud and supports macros for sending pre-typed strings.

850 EXPRESS

Keith Ledbetter, author of 1030 EXPRESS and now 850 EXPRESS has allowed ICD to distribute his latest terminal program with our P:R: Connection and R-Time 8. This is a fantastic terminal program written in ACTION! from OSS. 850 EXPRESS is worth more than most terminal programs you would pay $30 or more for in a store!

AMODEM7 and 850 EXPRESS are distributed on a "freeware" basis which means: Try the program out, if you like it and use it as your main terminal software, send the author payment of whatever you feel it is worth. (Send $5, $10, $20, etc.) You are free to distribute this freeware to your friends as long as you pass on this message and do not remove or modify the author's name, address, copyright notice, etc. from the program.

RSCOPE

Joe Miller originally wrote TSCOPE as a terminal program to work with the Atari 850 and COMPUSERVE's unique file transfer protocol. TSCOPE quickly became the standard terminal program for Atari COMPUSERVE users. Recently COMPUSERVE has added XMODEM protocol which has allowed users a greater choice of software. RSCOPE is a new 'R:' handler version of TSCOPE modified by Joe Miller to work with standard 'R:' handler devices and not just the Atari 850. We would like to thank Joe for his continuing support of the Atari 8-bit community.

PRC.SYS

Some programs (such as the current version of HOMETERM) may require our SIO emulation program called PRC.SYS. If using Atari DOS 2 or 2.5, copy PRC.SYS to a blank disk, rename it to AUTORUN.SYS and APPEND the AUTORUN.SYS from your terminal program to it (see your DOS manual). If using SpartaDOS, just put PRC.SYS in a batch file and run it first, before your terminal program.

ICD BBS (815) 968-2229

This is a good place to test out your new interface or MODEM. We support 300, 1200, 2400 baud communications and are in operation 24 hours a day, 7 days a week. No password is required for UPLOAD, DOWNLOAD or full message base access and there are no charges (other than long distance) to use this board. We do request that you use your real name and location when signing on.

Using the P:R: Connection with a Printer

If using a parallel printer, plug your printer cable from the parallel port (15 pin) of the P:R: Connection into your printer.

If using a serial printer, plug your printer cable from serial port 1 or 2 (9 pin) of the P:R: Connection into your printer. Since the Atari operating system defaults to a parallel printer, you must use a DOS (such as SpartaDOS from ICD) with the capabilities to divert all print output to the serial port or you must use programs which support serial printers.

Options

There are two user selectable hardware options inside the P:R: Connection. These are selected by opening the case and moving the jumper plugs at SW1 and SW2. If you already have a printer which supports the Atari without an interface (usually a direct connect Atari brand printer) you may want to use the P:R: Connection as a serial interface only. SW1 selects printer ON or printer OFF. Leave SW1 at the default of "P-ON" unless your direct connect printer does not function properly with the P:R: Connection installed.

If you test your printer out and it prints one line on top of another without feeding any paper, it needs a line feed for every carriage return. You can fix this by moving SW2 to "LF/CR". The default for SW2 is "CR only" which matches the Atari 850.

CHAPTER 2—THE PARALLEL AND SERIAL INTERFACES

The Parallel Interface

The parallel interface contains all the lines necessary to control standard parallel printers. Most parallel printers will use a 36 pin centronics connector. The signals listed in table 2-1 are supported by the P:R: Connection.

TABLE 2-1 Standard Parallel Printer Signals

Direction	Function	Pin
from P:R:C	Data Strobe	1
to P:R:C	Busy	13
to P:R:C	Fault	12
(none)	Data Pull up	9
(none)	Ground	11
from P:R:C	Data Bit 0	2
from P:R:C	Data Bit 1	3
from P:R:C	Data Bit 2	4
from P:R:C	Data Bit 3	5
from P:R:C	Data Bit 4	6
from P:R:C	Data Bit 5	7
from P:R:C	Data Bit 6	8
from P:R:C	Data Bit 7	15

The Serial Interface

The serial interface is RS-232-C compatible which means that you may connect any RS-232-C device to the P:R: Connection and communicate with it. There are actually two serial ports on the P:R: Connection. Port 1 ('R1:') is a full port which contains all necessary handshaking lines that some MODEMs and other devices require, and port 2 ('R2:') is a stripped port containing only the receive and transmit lines (the DTR and RTS lines are held in the "ready" state if needed). The P:R: Connection **does not** include a current loop port like port 4 of the old Atari 850. We felt that would add unnecessary expense since current loop interfaces are rarely used.

RS-232 Defined

The RS-232-C standard defines about 20 lines, of which, only about 8 are commonly used. Even though a device does not support all signals, it is still considered "RS-232 compatible". The P:R: Connection serial port 1 supports the signals listed in table 2-2. This port 1 matches port 1 on the Atari 850.

The P:R: Connection is considered a data terminal (also DTE or Data Terminal Equipment) whereas a MODEM is a data set (also DCE or Data Computer Equipment). There is no problem in connecting "data terminals" to "data sets", however, when connecting two "data sets" (or "data terminals"), you must take care since the signals are directional (i.e. you must cross XMT to RCV, DTR to CTS, etc.).

TABLE 2-2 The Most Common RS-232 Signals

Direction	Description	Abbreviation	Pin
from P:R:C	Transmitted data	XMT	3
to P:R:C	Received data	RCV	4
from P:R:C	Data terminal ready	DTR	1
to P:R:C	Signal (carrier) detect	CRX	2
to P:R:C	Data set ready	DSR	6
from P:R:C	Request to send	RTS	7
to P:R:C	Clear to send	CTS	8
(none)	Signal ground	GND	5

CHAPTER 3—CONCURRENT I/O VS. BLOCK MODE (RS-232)

Throughout this manual, numerous references are made to "concurrent I/O" and "block mode". These are simply two different methods of implementing the Atari serial bus for the transmission of serial data. **If you are to write programs supporting the P:R: Connection 'R:' Handler, you MUST understand the difference and what limitations each method presents.**

Block Mode. . . What is it?

Block mode is very much like reading or writing disk sectors. The data is saved in a buffer until either 1) the buffer is full, 2) an end-of-line character is placed in the buffer, or 3) the channel is closed. When one of these conditions is met, the entire buffer is transmitted from the computer to the P:R: Connection. This leaves the serial bus free for the computer to communicate to other devices.

There are two very serious limitations of block mode operation. The first being that input from the ports is not possible, thus block mode is output-only. Any input to the RS-232 port is simply ignored since the P:R: Connection does not store any data at its ports.

The second limitation is that data arriving at the RS-232 outputs is not "real-time". When simply sending data to another computer, a printer, or some other non-interactive peripheral, this mode of operation is sufficient. Data at the output will normally appear one line at a time.

Concurrent Mode I/O

While in concurrent mode I/O, the P:R: Connection simply acts as a bit carrier. In essence, it throws a switch connecting a port to the serial bus of the computer. Thus the serial device (POKEY) of the computer acts as a UART (universal asynchronous receiver transmitter).

In this mode, communication is full duplex (bi-directional) and occurs in "real-time". Thus, when in a terminal program, data you type appears at the output as you type it (unless you type faster than the current baud rate, in which case the data you type is buffered). A terminal program simply acts as a switch carrying data you type to the RS-232 handler output and the RS-232 handler input to the screen output handler.

Since the serial port is strictly used to carry port data, the serial bus may not be used for anything else while in concurrent mode. This means that neither printers nor disk drives may be active during concurrent mode. Instead, you must first close the RS-232 port and then perform the necessary disk I/O (or printing). The major drawback is that when the port is closed, any data arriving at the port will be lost—this is a problem with all serial RS-232 and MODEM interfaces for the Atari (this would not be a problem with a properly designed interface connected through the parallel expansion bus using a UART). Note that all Atari terminal programs do use concurrent mode I/O—they could not receive data if they didn't.

CHAPTER 4—RS-232 HANDLER FUNCTIONS AND TABLES

The following is a list of all input/output and XIO calls to the RS-232 ports of the P:R: Connection. Note that IOCB is an input/output channel number that indicates what OPEN device shall receive or provide data. For most XIO calls, you may use any legal IOCB number as long as it is NOT open to any other device. From Atari BASIC, you may use IOCB numbers 1 through 7 (0 is reserved for editor 'E:' I/O).

Note that IOCB #7 is used for the BASIC LPRINT statement and IOCB #6 is used for graphics modes functions from BASIC. Also if using SpartaDOS, IOCB #4 and IOCB #5 are used while doing output and input redirection respectively (via the DOS PRINT command and batch files).

'Rn:' is the serial interface port number being opened or used. For the P:R: Connection, 'n' can be either 1 or 2. The first is the full port (with all the handshake lines) of the P:R: Connection. Note that if you use 3 or 4 for 'n', ports 1 or 2 will be accessed rather than receiving an error.

All the function formats are given in their Atari BASIC form. If using assembly language of some other high level language, refer to the language manual for its equivalent form.

Opening an RS-232 Port

Syntax
OPEN #IOCB,Aux1,0,"Rn:"

Remarks
This function opens a channel to an RS-232 port in non-concurrent mode. This means that you may only input data after performing a start concurrent mode function (XIO 40). Note that Aux1 contains the I/O direction bits—4 for input only, 8 for output only, and 12 for both input and output (which is equivalent to 13 of the 850 interface). Many XIO calls do not require that you open an RS-232 channel first, however, it is good practice to open the channel first.

When a channel is opened, the buffer pointers are cleared for only the direction(s) in which the port is being opened. For example, if you are in concurrent mode I/O on port 1 using IOCB #2 and an open for output is performed on port 1 using IOCB #3, the data waiting in the input queue of port 1 is not lost. Multiple OPENs to the RS-232 ports have no effect on concurrent I/O. Thus, in this case, the system remains in concurrent I/O to port 1. In fact, if the second OPEN was for input (or both input and output), this channel would inherit the concurrent I/O characteristic of the first channel.

It is very important to understand the difference between concurrent and block mode for efficient and problem-free programming. Many XIO functions may only be performed during block mode (non-concurrent), however, input may only be performed during concurrent mode I/O. This is due to limitations of the Atari serial port.

Closing an RS-232 Port

Syntax
CLOSE #IOCB

Remarks
This statement will close the IOCB connected to the port in which a prior OPEN statement initiated. If another IOCB is connected to the same port, that connection will remain intact (data input buffers will not be lost). A CLOSE always flushes the data awaiting transmission (in the buffer) to the port indicated by the paired OPEN statement (OPEN prior to the CLOSE on the same IOCB).

Note that the CLOSE will shut down any concurrent I/O even if another IOCB is open to a port. This is usually relevant only when two IOCB's are open to the "Rn:" device. For example, suppose IOCB #1 is open for input on port 1 (in concurrent mode), and IOCB #2 is open for output on port 2. A CLOSE on port 2 will disable the concurrent mode of port 1 thus requiring another XIO 40 to re-enable concurrent I/O. This operation also causes an error since port 2 did not have control over the serial bus. If the CLOSE were performed on port 1, no error would occur, but concurrent mode is still disabled. Thus, the only way to terminate concurrent I/O properly is to CLOSE an IOCB opened to the port currently in concurrent mode. (It is possible to have two IOCB's opened to the same port—concurrent I/O is a property of the connection to the port rather than of the IOCB. An IOCB number simply establishes a reference number (IOCB #) to a port.)

Input Character or Line From RS-232 Port

Syntax
GET #IOCB,varb
INPUT #IOCB,varb$

Remarks
These functions input data from the RS-232 port specified by a preceding open statement. The GET statement inputs the numeric value of one character into a numeric variable. The INPUT statement inputs a string of characters into a string variable. If the input is a numerical ASCII string, you may input into a numeric variable. Input strings are terminated by an end-of-line (EOL) character.

Note that the IOCB must be opened for read or read/write and you must be connected to the port (as indicated by open) in concurrent mode. If you are not in concurrent mode to the correct port, an input attempt will shut down the other port's concurrent I/O. Refer to your BASIC reference manual for more information.

Output Character or Line To RS-232 Port

Syntax
PUT #IOCB,exp
PRINT #IOCB;exp$

Remarks
These functions output data to the RS-232 port specified by a preceding open statement. The PUT statement outputs the numeric value of one character to the port, and the PRINT statement outputs a string of characters to the port. The syntax of the PRINT statement is the same as a normal PRINT statement except that the "#IOCB;" precedes the expression.

Note that the IOCB must be opened for write or read/write but you do not have to be connected to the port (as indicated by open) in concurrent mode. Refer to your BASIC reference manual for more information.

Reading the Port Status

Syntax
```
STATUS #IOCB,DUMMY
FLAGS = PEEK(746) : REM Error bits relating to status history
LINESTAT = PEEK(747) : REM Status of handshake lines
or
STATUS #IOCB,DUMMY
FLAGS = PEEK(746) : REM Error bits relating to status history
INCHARS = PEEK(747) : REM Number of chars in input buffer
OUTCHARS = PEEK(749) : REM Number of chars in output buffer
```

Remarks

These statement sequences are useful for determining many facts about the state of the RS-232 ports. The first syntax is used when in block mode I/O, whereas the second is used in concurrent mode I/O. Notice that the variable DUMMY is simply a CIO status of the success of the STATUS command. If there were an error (DUMMY<>1), then BASIC would halt and give an error message (unless a TRAP was performed prior to the STATUS).

The block mode STATUS (first syntax) returns a status history of the port (in FLAGS) and the state of the control lines (in LINESTAT). The meaning of each bit is given in tables 4-1 and 4-2.

The concurrent mode STATUS (second syntax) returns a status history of the port (in FLAGS) and the number of characters in the input buffer (in INCHARS) and in the output buffer (in OUTCHARS). The meaning of each bit of FLAGS is given in table 4-1.

TABLE 4-1 Meaning of Error Bits From Location 746

Bit Number	Decimal Equiv.	Error Meaning
7	128	Received a data framing error
6	64	Received a data byte overrun error
5	32	Recieved a data parity error
4	16	Received a buffer overflow error (> 255 chars)

TABLE 4-2 Meaning of Status Bits From Location 747

Bit Number*	Decimal Equiv.	Meaning When Bit is Set (1)
7	128	DSR is true (ready)
5	32	CTS is true (ready)
3	8	CRX is true (ready)
0	1	RCV is at MARK (high state)

*Bits 6, 4, and 2 are simply copies of the next highest bit. In the 850 Interface, these bits would indicate a history (i.e. not always ready since last STATUS).

Forcing Early Transmission of Output Blocks

Syntax
XIO 32,#IOCB,0,0,"Rn:"

Remarks
This function causes all the buffered data in the computer to be outputted to the RS-232 port. This works for either block or concurrent mode. Note that if in concurrent mode, bytes are put in a buffer, not to the port directly. The data is then taken out of the buffer and sent to the port when the last byte sent is finished. Thus, you can send data to the CIO (by PRINT, or PUTs) faster than it is transmitted out of the computer.

When an RS-232 port is closed (see CLOSE statement), the data in the buffer is not lost; transmission of the remaining data is forced.

Controlling Outgoing Lines DTR, RTS, and XMT

Syntax
XIO 34,#IOCB,Aux1,0,"Rn:"

Remarks
This function allows you to set the state of the output handshaking lines. This function may not be used while in concurrent mode (see "Setting Concurrent Mode"). Aux1 is coded as indicated by table 4-3.

TABLE 4-3 Control Values Added to Aux1 (XIO 34)

Function	Bit	Decimal Equiv.	Meaning When Bit is SET
DTR	7	128	Set state of DTR (from bit 6)
	6	64	Set DTR Ready (Not ready if bit is CLEAR)
RTS	5	32	Set state of RTS (from bit 4)
	4	16	Set RTS Ready (Not ready if bit is CLEAR)
XMT	1	2	Set state of XMT (from bit 0)
	0	1	Set XMT to MARK (SPACE if bit is CLEAR)

Setting Baud Rate, Stop Bits, and Ready Checking

Syntax
XIO 36,#IOCB,Aux1,Aux2,"Rn:"

Remarks
This function configures the RS-232 port for desired speed and stop bits. It also tells the port which handshake lines to monitor. This function should be used *before* entering concurrent mode (XIO 40), since it may not be used while *in* concurrent mode (see "Setting Concurrent Mode").

Aux1 is the sum of two codes; baud rate and the number of stop bits. The coding is given by Table 4-4. You must add the value representing the desired baud rate to the code (0 or 128) for the desired number of stop bits per word. Note that the word size is always 8 bits plus 1 or 2 stop bits; the P:R: Connection does not support smaller word sizes as did the Atari 850 interface.

Aux2 is coded to be the sum of 3 values (as given by table 4-5). Each value represents a control line to monitor. If the value is 0, then that control line is not monitored. The handshake lines are only checked when you enter into concurrent I/O mode.

TABLE 4-4 Codes to Add to Aux1 (XIO 36)*

Add	Baud Rate	Add	Baud Rate
0	300	8	300
1	45.5	9	600
2	50	10	1200
3	56.875	11	1800
4	75	12	2400
5	110	13	4800
6	134.5	14	9600
7	150	15	19200

*Default is 1 stop bit. Add 128 for 2 stop bits.

TABLE 4-5 DSR CTS CRX Checking Codes for Aux2 (XIO 36)*

Bit	Add	To Check This Line (Before Sending/Receiving Data)
0	1	CRX
1	2	CTS
2	4	DSR

*Default is 0 which indicates no checking of handshake lines.

Setting Translation Modes and Parity

Syntax
XIO 38,#IOCB,Aux1,Aux2,"Rn:"

Remarks
This function configures the input and output parity and the level of ASCII/ATASCII translation. Aux1 is coded to specify all these parameters while Aux2 is the "won't translate" character. This character is only used in the "heavy ATASCII/ASCII translation" mode and is returned (during a GET or INPUT) when the incoming character is not an ASCII character with a value of 32 to 127 ($20 to $7F in HEX). The value of Aux1 is derived from table 4-6.

TABLE 4-6 Control Values Added to Aux1 (XIO 38)

Function	Add	Resulting Function Performed
OUTPUT PARITY	0	Do not change parity bit (default)
	1	Set output parity to odd parity
	2	Set output parity to even parity
	3	Set parity bit to 1
INPUT PARITY	0	Ignore and do not change parity bit (default)
	4	Check for odd parity, clear parity bit
	8	Check for even parity, clear parity bit
	12	Do not check parity, clear parity bit
TRANS-LATION	0	Light ATASCII/ASCII translation (default)
	16	Heavy ATASCII/ASCII translation
	32	No translation
LINE FEEDS	0	Do not append LF after CR (default)
	64	Append LF after CR (translated from EOL)

Setting Concurrent Mode

Syntax
XIO 40,#IOCB,0,0,"Rn:"

Remarks
This function starts concurrent mode I/O with RS-232 port 'n'. A successful OPEN statement must be performed before entering concurrent I/O. Note that you should also perform all other XIO (34, 36, and 38) statements before this statement. You must set concurrent I/O before any attempts to input data through the RS-232 port.

For more information on concurrent mode I/O, refer to Chapter 3 "Concurrent I/O vs. Block Mode".

APPENDIX A—OTHER ICD PRODUCT OFFERINGS

R-TIME 8 Clock Cartridge - $69.95 - adding a new dimension to the Atari: TIME/DATE—developed to provide continuous and automatic time/date information—plugs into any cartridge slot on an Atari Computer and updates SpartaDOS automatically and accurately. The R-Time 8 works in the right or left slot of the 800 computer. A unique cartridge extension port in the top allows the use of other cartridges when using an XL/XE series computer. The R-Time 8 is supported through software as the clock for *Bulletin Board Construction Set (BBCS)* from the ANTIC Arcade. Our new 'ZHAND' clock handler works with any DOS and allows simple XIO calls for easy R-Time 8 access through your favorite programming language. It comes with a built in 3-5 year battery! The R-TIME 8 does **not** use any cartridge area memory and is decoded in the $D5B8 - $D5BF range so it will not interfere with other cartidges including BASIC XE. ICD will provide complete service for $20 US flat rate which includes all parts, timing adjustment, and battery replacement.

SpartaDOS Construction Set - $39.95 - a special SpartaDOS utility package which includes seven different SpartaDOS versions and many new disk utilities including: *RAMDISK support for the 130XE and RAMBO XL*, ATR8000 support, RPM test, a high speed sector copier, hard disk drive support, and much, much more. A special menu file allows rapid transfer, erasure, lock or unlock of tagged files, using *only* the SPACE bar, OPTION, START, cursor and SELECT keys. You want more you say? How about a 32 character keyboard buffer, intelligent switching between disk densities, a binary file games menu, subdirectories, time/date file stamping? SpartaDOS is the *only* DOS with all these features. The new SDCS manual is 175 typeset pages of everything you wanted to know about SpartaDOS and the US Doubler. Also included is the SpartaDOS version 3.2 software and manual for BASIC XE, 1200XL, and hard disk drive support. There is so much software included in this package that we distribute SDCS on two unprotected disks using three sides! *Included free* are seven arcade games.

SpartaDOS, the essence of SDCS, is a totally new advanced DOS for the Atari Computer. SpartaDOS enhances the entire 8-bit Atari Computer line; gives it the power of an Apple or IBM machine. SpartaDOS puts a time/date stamp on each file when it is created or rewritten. This information is then displayed when the directory is listed. SpartaDOS also supports the 1050 "dual density" which means you can use it with your unmodified 1050 in both single and dual density. Our US Doubler allows your 1050 access to all three densities. SpartaDOS also works with the Indus GT, Rana 1000, and other third party drives for the Atari.

SpartaDOS is also memory resident. This means you can go to BASIC then back to DOS without reloading DOS from disk. Special XL/XE SpartaDOS versions even give you an extra 4000 bytes of free memory! Input and output redirection is supported through 'batch files' and the 'PRINT' command. SpartaDOS supports the ATR8000 with 5¼ and 8 inch, single or double sided drives, 36-80 tracks. An ATR8000 serial communications handler is also provided.

You get SpartaDOS *free* as part of the SpartaDOS Construction Set or with the US Doubler.

So you want a second or third opinion, then let's talk about reviews.

In his February 1986 ANALOG review of SpartaDOS, Matthew Ratcliff raved, "With this DOS, I can read from and write to any DOS in any density—without any special utilities. This single feature makes SpartaDOS the most powerful disk operating system I've ever seen for the Atari XL/XE Computer." In October 1985 ANALOG Computing, Art Leyenberger states in his column *The End User:* "There's an old saying that someone will always build a better mousetrap. ICD will certainly be catching more than mice with their new SpartaDOS Construction Set (SDCS). . .As I said before: the more I use it, the more I like ICD's SpartaDOS Construction Set. It could easily be the ultimate DOS for the 8-bit Atari computers." In May 1985 ANALOG Computing, Russell Haupert, in his review of the US Doubler from ICD, Inc., raved: "The DOS is very rich in features and a great pleasure to use." Peter Ellison in ROM Magazine (DEC/JAN 85): "SpartaDOS is the best I have ever used. . .a product so revolutionary it adds new life to the Atari. . ." What more can we say?

US Doubler! - $69.95 - True Double Density (180 kilobytes) and high speed I/O for the 1050 drive. Also supports single density and the 130 KB Atari 'Dual Density' or 1050 mode. This is fully compatible with all existing Atari software. The high speed I/O (UltraSpeed) works with SpartaDOS to *TRIPLE* the speed of data transfer! Now it includes a *free* SpartaDOS Construction Set. Two plug-in chips in the 1050 must be replaced and now, *no soldering* is required in most cases. (Call your dealer for details.) This hardware modification also fixes some of the bugs in the early 1050s that prevented them from loading certain software programs. Even when not using SpartaDOS, the US Doubler increases the 1050's speed by about 8% and reads and writes more accurately than most standard 1050s. The US Doubler makes your 1050 drive compatible with all other Atari DOS's but we are sure you will use SpartaDOS as your main DOS once you learn the natural and powerful command structure. The new US Doubler package includes the SpartaDOS Construction Set manual with an easy to follow installation section, two SpartaDOS program diskettes, and the plug-in hardware modules.

What do the reviews say, you sagely inquire?

In the MAY 1985 issue of ANALOG Computing, Russell Haupert waxes: "In tests, the UltraSpeed I/O worked as stated. . . The old bleep bleep of POKEY is replaced by a staccato rush that sounds more like machine gun fire. . . I've tested the US Doubler in all three formats and am happy to report that it performs as promised. . . Most importantly, using the true double density afforded by this enhancement I've attained compatibility with my friends' disks, as well as reducing my disk count by one half." Peter Ellison of ROM Magazine was equally impressed: "If you have a 1050, this is a fantastic deal. It is fully compatible with all software, and allows a whole lot more disk space." In the December 1985 annual shoppers guide, ANTIC, the Atari Resource, selected US Doubler as one of the *best products of 1985* in the Unique Hardware category.

US Doubler 5-8 - $49.95 - this was developed for registered US Doubler owners who desire to use more than four drives with their Atari computer. The most obvious use is for bulletin boards. This special US Doubler chip set allows your drive to be set to any drive number from 5 through 8. Neither SpartaDOS nor installation instructions are included since this is for registered US Doubler or SDCS owners only. It works with any DOS which supports more than 4 drives. For operation under SpartaDOS use version 2.3d or later. Note: you will *not* be able to use this drive as #1 (only 5-8) to boot the system with this special chip set installed.

RAMBO XL - $49.95 - makes your 800XL or 1200XL memory compatible with the 130XE! This internal memory upgrade kit supplies all the signals necessary to turn your XL into a powerful 256K computer. Now you can use powerful BASIC XE with your XL computer! RAMBO XL supports the standard 64K RAMDISK supplied with Atari DOS 2.5. The new RD.COM handler with SpartaDOS Construction Set gives a 192K RAMDISK! That's enough memory to duplicate a full double density disk in one pass! The new Synfile + database from Synapse/Broderbund now supports this memory expansion! RAMBO XL includes a plug-in decoding board and complete installation instructions. You supply the 8-256K DRAMs which plug into existing sockets where the 64K chips now reside (available for $32.00 direct, price may change). Note: Some 800XLs don't have sockets on all ICs. This will require extensive de-soldering and should only be attempted by a professional. ICD will install and test RAMBO XL on your computer for $30.00 including shipping.

Printer Cable - $15.00 - between your Printer and P:R: Connection. Cables are necessary to connect your printer to the P:R: Connection. High quality 6 foot cables are available for printers with a 36 pin 'centronics' type connector. Other configurations available upon request.

MODEM Cable - $15.00 - between your MODEM and P:R: Connection. Cables are necessary to connect your MODEM to the P:R: Connection. High quality 6 foot cables are available for MODEMs with the standard DB25S. Other configurations available upon request.

Be sure to register your P:R: Connection by filling out and sending in your warranty/update card at the back of this manual. This will make you eligible for a *free* subscription to our quarterly newsletter, OPEN FILE! This will help to keep you abreast of our latest products and newest designs.

APPENDIX B—P:R: CONNECTION SIO COMMANDS

For the sake of compatibility and interests of all who use the P:R: Connection, as much technical information is included in this manual as possible. In this appendix, all SIO commands available to the P:R: Connection are given. We encourage you to use this information to make the P:R: Connection a mainstay in the Atari market.

SIO Commands for the Serial Interface (for R: Handlers)

On all SIO commands, the RS-232 port number is encoded into the device ID; a $50 port is 1, and $51 is port 2 (this is calculated by SIO as DEVIC + DUNIT-1). The device commands (DCOMND) are listed below followed by their function. Note that AUX1 and AUX2 are copies of memory locations $30A and $30B respectively (normally the sector number). The data direction is determined by DSTAT (location $303) where $80 indicates output (from computer) and $40 indicates input (to computer).

A($41) = Set state of DTR/RTS/XMT lines
No data frame
AUX1 = Coded data as follows
Bit[0] : New state of XMT (0 = SPACE)
Bit[1] : 1 if to set new state of XMT, 0 if no change
Bit[4] : New state of RTS (0 = OFF)
Bit[5] : 1 if to set new state of RTS, 0 if no change
Bit[6] : New state of DTR (0 = OFF)
Bit[7] : 1 if to set new state of DTR, 0 if no change

S($53) = Get state of CTS/CRX/DSR lines
Data frame returned (4 bytes):
+1 = Returned status coded as follows:
Bit[0] : Current state of RCV, 1 = MARK, 0 = SPACE
Bit[1] : (same as B0)
Bit[2] : (same as B3)—no history given
Bit[3] : Current state of CRX, 1 = ready (on line)
Bit[4] : (same as B5)—no history given
Bit[5] : Current state of CTS, 1 = ready
Bit[6] : (same as B7)—no history given
Bit[7] : Current state of DSR, 1 = ready

X ($58) = Enter concurrent mode
No data frame
AUX 2 = Index of lines to monitor coded as follows:
Bit[0] : 1 = Check CRX line ready—NAK returned if not ready
Bit[1] : 1 = Check CTS line ready—NAK returned if not ready
Bit[2] : 1 = Check DSR line ready—NAK returned if not ready
To exit concurrent mode, pulse COMMAND low for at least 100uS. The P:R: Connection is fast enough to react to the command (if any) that caused the COMMAND to be pulsed, however, the standard P:R: Connection handler simply pulses COMMAND low with no command frame being sent.

? ($3F) = Get parameters of boot segment
Data frame returned (12 bytes)
+0 = 12 bytes of data to put in DCB for next SIO call

! ($21) = Get boot code segment from P:R: Connection
Data frame returned (#bytes determined by '?' command)
+6 = Run address to finish RS-232 handler load process

% ($25) = Main handler transmission command
Data frame returned (#bytes as used in boot code)
+0 = start of RS-232 handler code

Note that there is no write command. To output data in block mode, you must first enter concurrent I/O and then send the data as per concurrent mode. When transmission is finished (last character emptied from buffer), you should wait a few jiffies and then shut down concurrent I/O.

APPENDIX C—R: HANDLER SOURCE CODE

This appendix contains the source code of the 'R:' handler of the P:R: Connection. Lately, it has become a trend to include a 'R:' handler that supports several devices (e.g. an RBIN handler has been around for some time that supports the 850 interface, 1030 MODEM, 830 MODEM, and the XM301 MODEM). The P:R: Connection is similar to the 850 interface on the SIO level, however, a few key SIO calls are lacking; they are 'B', for set baud rate, and 'W', for write block. These are not needed by the P:R: Connection since it emulates block mode by 1) entering concurrent mode, 2) sending the data, and 3) exiting concurrent mode. But, as a result of the missing SIO commands, the RBIN handler is not compatible with the P:R: Connection.

Generally, the authors of terminal programs and BBS's (bulletin board systems) allow an 'R:' handler to be loaded before the program loads. Thus, most of these programs will work with the P:R: Connection. The one most notable exception is HOMEPACK (version 1). This program has special XIO calls which emulate the suspend and resume functions of the Atari MODEMs (which use a T handler). Therefore, a special 'R:' handler called PRC.SYS is supplied on our distribution diskette.

We are making every effort to insure that the P:R: Connection will remain compatible with all communications software. If you should find problems and/or incompatibilities with the P:R: Connection, please don't hesitate to call the ICD BBS and leave your comments or questions. Thanks to everyone for making our product a success.

```
        title   'R: Handler for P:R: Connection -- Appearence after Installing'
                '©1986 ICD, Inc.'

;       Atari SIO interface
;       -------------------
ddevic  equ     $300            ; device ID
dunit   equ     $301            ; device unit number
dcomnd  equ     $302            ; SIO command
dstat   equ     $303            ; SIO status
dbuflo  equ     $304            ; data buffer low
dbufhi  equ     $305            ; data buffer high
dtimo   equ     $306            ; device timeout value
dbytlo  equ     $308            ; number of bytes low
dbythi  equ     $309            ; number of bytes high
daux1   equ     $30A            ; auxilary 1
daux2   equ     $30B            ; auxilary 2
sio     equ     $E459           ; SIO

intvec  equ     $20A            ; SIO interrupt vectors (3)
imirqv  equ     $216            ; Immediate IRQ vector


;       Atari CIO interface
;       -------------------
icdnoz  equ     $21             ; zp device number
iccomz  equ     $22             ; zp command
icax1z  equ     $2A             ; zp aux1 (direction/XIO info)
icax2z  equ     $2B             ; zp aux2 (XIO info)
hatab   equ     $31A            ; handler table

tchar   equ     $43             ; these locations also used by DOS
portn   equ     $44
iocb    equ     $45


;       CIO errors for R: handler
;       -------------------------
ilcom   equ     $84             ; illegal command
noper   equ     $85             ; not open
enack   equ     $8B             ; NACK
porop   equ     $96             ; Port open
inconc  equ     $99             ; in concurrent mode


;       Atari system hardware registers
;       -------------------------------
pokmsk  equ     $10             ; IRQ enable shadow
irqen   equ     $D20E           ; IRQ enable register
irqst   equ     $D20E           ; IRQ status
brkflg  equ     $11             ; break flag
skctl   equ     $D20F           ; Pokey I/O control
audctl  equ     $D208           ; Audio channel pairing register
skstat  equ     $D20F           ; Pokey I/O status
skres   equ     $D20A           ; Pokey reset
serinr  equ     $D20D           ; serial input
seroutr equ     $D20D           ; serial output
pbctl   equ     $D303           ; Port B control (command line)
audf1   equ     $D200           ; audio freq 1-4/control 1-4
audc1   equ     $D201
audf2   equ     $D202
audc2   equ     $D203
```

```
audf3   equ     $D204
audc3   equ     $D205
audf4   equ     $D206
audc4   equ     $D207

dosvec  equ     $0A
inivec  equ     $0C
jiffy   equ     $14             ; jiffy LOW counter
memlo   equ     $2E7            ; Low memory ptr
stloc   equ     $2EA
bptr    equ     $43
ssflag  equ     $2FF


        sbttl   'Initialization and SIO routines'
        page

;       Initialize after reset
;       ----------------------

initz   bit     xwarm           ; if warm, then initz rest
        bmi     resi
xinit    jsr    $ffff           ; do not perform DOS INI on first initz
                                ;  of RS-232 handler... this is because if
                                ;  this is AUTORUN.SYS, it is loaded by DOS INI
                                ;  code... can't go recurse
                                ; NOTE: The $FFFF is replace by the contents of
                                ;  DOSINI by some initz code... DOSINI then pts
                                ;  to INITZ... ex:
                                ;       lda dosini
                                ;       sta xinit+1
                                ;        lda dosini+1
                                ;        sta xinit+2
                                ;       lda #low initz
                                ;       sta dosini
                                ;        lda #high initz
                                ;        sta dosini+1
                                ;       jmp (dosini)
resi    ldy     #0
        sty     xwarm           ; from now on, always initz DOS
        lda     #low cend       ; set memlo to END of handler
        sta     memlo
         lda    #high cend
         sta    memlo+1

cha     lda     hatab,y
        beq     gotnp           ; jump if entry is free
         cmp    #'R'
         beq    docls           ; jump if already present
          iny
          iny
          iny
         bne    cha             ; check next entry.. assume that
                                ;  there is a free entry
gotnp   lda     #'R'            ; create new 'R' entry in table
        sta     hatab,y
         lda    #low rhand
         sta    hatab+1,y
          lda   #high rhand
          sta   hatab+2,y

docls   lda     #$80
```

```
        ldy     #10
wx7     sta     conflg-1,y      ; clear CONFLG,INP,INEND
         dey                    ;  OUP,OUEND,NOCHARS
         bne     wx7
         iny                    ; set Y to 1.
        rts

hiad    db      (obuf)/256
        db      (obuf+256)/256  ; table of high addresses of OBUF


;       Set concurrent mode for interface
;       ---------------------------------

set.con sta     dcomnd
         ldy    #0
         sty    dstat
         iny
        iny
        sty     dtimo
        jmp     sio


        sbttl   'Handler vectors and command entry points'
        page

;       RS232 device handler
;       --------------------

rhand   dw      ropen-1         ; OPEN
        dw      rclose-1        ; CLOSE
        dw      rget-1          ; GET BYTE
        dw      rput-1          ; PUT BYTE
        dw      rstatus-1       ; STATUS
        dw      rspec-1         ; XIO

rvecs   dw      sirdy           ; Serial input ready interrupt
        dw      sordy           ; Serial output ready interrupt
        dw      socmp           ; Serial output complete


;       save unit number in table
;       -------------------------
; in:
;       X       = iocb number ($00, $10,...)
; out:
;       X       = iocb index (0,1,...)
;       Y       = port number (0 or 1)

                                ; Entry for getting unit number
                                ;  with use for put and get
getun   txa                     ; get unit number
        lsr     a
        lsr     a
        lsr     a
        lsr     a
        tax
        lda     unit,X          ; get unit number
         bcc    setun           ; go put in correct places (jump always)

retri   txa                     ; get iocb index
        lsr     a
```

```
        lsr     a
        lsr     a
        lsr     a
        tax
        ldy     icdnoz          ; get unit number from iocb
        dey
        tya
        and     #1
setun   stx     iocb
        sta     dunit           ; set unit number for SIO
        sta     unit,x          ; save in unit table
        sta     portn           ;  set port number
        tay
         lda    #$51
         sta    ddevic
        rts


;       Command entry points
;       ====================

;       OPEN COMMAND
;       ------------

ropen   jsr     retri           ; save unit in IOCB table
        lda     icaxlz          ; save direction bits
        sta     iodir,x
        ldx     #0
        stx     errflg          ; clear error flag
                                ; reset input/output ptrs
        lsr     a               ;  depending if read/write
        lsr     a
        lsr     a
        bcc     cwrt
         sty    inp
         sty    inend
cwrt    lsr     a
        bcc     oprt
;        ldy    portn           ; note that initz values of ptr doesn't matter
;        lda    #0              ; Y is portn from retri...
         sta    oup,y
         sta    ouend,y
oprt    ldy     #1
retx    rts                     ; return good status


;       CLOSE COMMAND
;       -------------

rclose  jsr     getun           ; get unit number etc.

;       Flush output buffer (alias RCLOSE)
;       ----------------------------------
; notes:
;       If in concurrent mode (on ICDNOZ) then just wait till all is
;       transmitted.  If not in concurrent, start concurrent and flush
;       buffer.  If in concurrent on the other channel, then abort with
;       an error (and take it out of concurrent).

flush   jsr     modck           ;place in concurrent mode
         bmi    retx            ;  exit if error (simple rts)
        jsr     enaoi           ;enable output IRQ
```

```
flus2   lda     brkflg
        beq     rstp
         ldy    portn           ; get port number to flush...
         lda    nochars,y
         beq    flus2

        lda     jiffy
        adc     #20
wtjif   cmp     jiffy           ; wait 20 jiffies (this should probably be
        bne     wtjif           ;  less... should be 1/30 sec+)

        ldy     #1              ; successful operation
         db $2C
rstp    ldy     #$80            ; break error.
;       jmp     restor          ; fall through to restor


;       Restore after concurrent mode
;       -----------------------------
; notes:
;       This must not affect the Y register.

restor  bit     conflg          ; only restore if not
        bmi     xrest           ;  already in concurrent mode

        sei
        lda     #$34            ; set command line low..
        sta     pbctl

        lda     pokmsk
        and     #$C7            ; disable input and output interrupts...
        sta     pokmsk          ;  Prev. only disabled input -- may have
        sta     irqen           ;  been cause for crashes when break pressed...

        ldx     #6-1            ; restore Pokey IRQ vectors
rslpl   lda     tpok,x
        sta     intvec,x
         dex
         bpl    rslpl

        ldx     #8-1
        lda     #0
lpla    sta     audf1,x         ; turn all sound off
         dex
         bpl    lpla

        lda     #$80
        sta     conflg          ; not in concurrent mode

        lda     #$3C
        sta     pbctl           ; COMMAND line HIGH
        CLI
xrest   rts


;       Enable output IRQ
;       -----------------
; notes:
;       It is ok to do this if no chars in buffer. The interrupt will
;       simply detect an empty buffer and disable the output IRQ.

enaoi   lda     #8              ; set 2 or 1 stop bits
```

```
        bit     baudr
        bmi     sesb
         lda    #$18
sesb    ora     pokmsk
        sta     pokmsk
        sta     irqen
        rts


;       Status command
;       --------------

rstatus jsr     retri           ; get unit number etc.
        bit     conflg          ; in block mode.. get ctr lines
         bpl    incomo          ;   jump if in concurrent mode

        ldy     #8-1            ; set up DCB
lp3     lda     sttab,y         ; setup DCB for status comnd
        sta     dcomnd,y
         dey
         bpl    lp3
        jsr     sio             ; do serial I/O
        bpl     seer            ; finish setting up error flags
         bmi    ret1b           ;  (same as return) return if error

incomo  sec
        lda     inend
        sbc     inp
        sta     stloc+1
         sec
;       ldy     portn           ; Y is portn from retri
         lda    ouend,y
         sbc    oup,y
         sta    stloc+3

seer    lda     errflg          ; get error flags
        sta     stloc
         ldy    #0
         sty    stloc+2
         sty    errflg          ; reset error flags
         beq    great           ; return good status


;       Special XIO command
;       -------------------

rspec   jsr     retri           ; get unit from iocb
        lda     iccomz          ; get command
        cmp     #40             ; check if to set concurrent mode
        bne     compa


;       Set concurrent I/O
;       ------------------
; notes:
;       If alreay in concurrent mode, an error will occur and concurrent
;       mode will be disabled.

        lda     conflg          ; make sure not alreay in
        bpl     incmod          ;  take it out of conc mode if error
         jsr    modck           ; make sure in concurrent
ret1b    clc
```

```
         bcc     return          ; exit with given status...

                         ; ------------------------------------
compa    cmp     #32             ; force write of short block
         bne     compb


;        Force write of short block
;        --------------------------
; notes:
;        This simply does a flush on the channel provided. The same
;        errors that occur on flush can occur here. If in concurrent mode,
;        an error should occur.

         lda     conflg          ; check if in concurrent
         bpl     incmod          ; jump if already in concurrent mode.
          jsr    flush           ; do flush without setup
          db $2C                 ; skip the good status return
great    ldy     #1              ; return with good status
return   ldx     iocb            ; get device ID
         lda     iodir,x         ; restore IO direction
         sta     icax1z
         rts

incmod   jsr     restor          ; take out of conc mode
         ldy     #inconc         ; illegal op while in conc mode
         bne     return          ; return with error

                         ;------------------------------------
compb    cmp     #34             ; check if to set control lines
         bne     compc


;        Set control lines (XIO 34)
;        --------------------------

         bit     conflg          ; jump if in block mode
         bmi     inblk

          lda    icax1z          ; get mode
          and    #3              ; XMT setting
          lsr    a               ; BIT[0]=0 if no change
          beq    great           ; exit with good status if no change
           lda   #$73            ; get break/no break
           bcs   setb
            ora  #$80
setb       sta   skctl           ; set break/no break
grtj2    clc
         bcc     great

inblk    lda     icax1z
         sta     daux1           ; set aux 1 byte
         lda     #'A'
         jsr     set.con         ; go set state of ports
          clc
          bcc    return          ; return with status from SIO


                         ; ------------------------------------
compc    cmp     #36             ; check if to set baud rate...
         bne     compd
```

```
;       Set baud rate (XIO 36)
;       ----------------------
; notes:
;       This operation is always legal. It can change baud rate right
;       in the middle of concurrent I/O if it wants.

        lda     icax1z          ; get baud rate/word size/stop bits
        sta     baudr,y
         lda    icax2z          ; get CTS flag
         and    #7
setct    sta    ctsflg,y
        clc
        bcc     grtj2           ; return with good status

                        ; ---------------------------------
compd   cmp     #38             ; check if to set translation mode
        bne     undef


;       Set translation (XIO 38)
;       ------------------------
; notes:
;       This operation is also always legal. Translation mode may be changed
;       right in the middle of concurrent mode if desired.

rtrans  lda     icax1z          ; get translation mode
        sta     tramod,y
         lda    icax2z          ; get won't translate charac
         sta    trachr,y
        clc
        bcc     grtj2

                        ; ---------------------------------
undef   ldy     #ilcom          ; illegal command error
        rts


;       Place handler into concurrent mode if not already
;       -------------------------------------------------
; notes:
;       This routine first checks to see if it is in concurrent mode,
;       if so, then it makes sure the right port is opened. If not, it
;       will place itself into concurrent mode. Errors are a SIO type
;       error if P:R: is not on, or a already concurrent type of error.

modck   lda     conflg          ; check if concurrent mode
        bmi     ctok            ; go make sure its the same

        cmp     portn           ; make sure same as device ID
        bne     incmod
         ldy    #1              ; return with good status...
reter1   rts

                                ; Set concurrent mode (unconditionally)
                                ; -----------------------------------
ctok    ldy     portn
        lda     ctsflg,y
        sta     daux2           ; controls must be ready for X command
        lda     #'X'
ctok2   jsr     set.con
         bmi    reter1          ;  jump if an error occurs...
```

```
        ldy     portn           ;  get port number (0 or 1)
        sty     conflg          ; now in concurrent mode
        lda     baudr,y         ; get baud rate
        and     #$0F            ; strip other bits..
        tay
         lda    bauth,y         ; get baud rate HIGH
         sta    audf2
         sta    audf4
        lda     baut1,y         ; get baud rate LOW
        sta     audf1
        sta     audf3

        lda     #$A0            ; set no sound..
        sta     audc1
        sta     audc2
        sta     audc3
        sta     audc4

        SEI                     ; disable IRQ while setting up
        lda     #$73            ; set SKCTL..
        sta     skctl
         lda    #$78
         sta    audctl          ; set channel pairing

        ldy     #6-1            ; get new IRQ vectors..
seir2   lda     intvec,y        ; save old vectors
        sta     tpok,y
         lda    rvecs,y
         sta    intvec,y
          dey
          bpl   seir2

        lda     #$20            ; enable input interrupt...
        ora     pokmsk          ; set pokey mask bits
        sta     pokmsk
        sta     irqen
         CLI                    ; enable interrupts..
         ldy    #1
         rts


;       Put byte
;       --------

rput    sta     tchar           ; save character
        jsr     getun           ; get unit number from table

        lda     tramod,y        ; Get translation mode
        and     #$30
        tay
         cmp    #$20            ; jump if no translation
         bcs    cpari

        lda     tchar           ; check if EOL
        cmp     #$9B
        bne     ckpar
         lda    #13             ; if EOL send CR/LF
         jsr    sench           ;  send char to buffer
         tya
          bmi   expu
```

```
        ldx     portn
        lda     tramod,x        ; check if to append LF
         asl    a
         bpl    expug           ; exit with good status
          lda   #10             ; send LF
          bne   sench

ckpar   and     #$7F            ; check if valid character
        dey
        bmi     sench
         cmp    #' '            ; heavy translation
         bcc    expug
         cmp    #$7D
         bcc    sench           ; jump if valid character
expug   ldy     #1
expu    rts

sench   sta     tchar           ; save character
cpari   ldx     portn
cpar    lda     tramod,x        ; do parity build
        and     #3
        beq     sencm
        asl     tchar
         cmp    #3
         beq    setpl           ; set high parity
        lsr     a
        lda     tchar
pupa    bcc     shlpp
         eor    #$80            ; (odd parity)
shlpp   asl     a
        bne     pupa
setpl   ror     tchar           ; set parity bit

sencm   ;ldx    portn           ;get device number... (already loaded)
        lda     hiad,x          ; set address of buffer
        sta     wx49+2          ; stuff it...
        ldy     ouend,x         ; get ptr to end of output
        iny
        tya

wpu     jsr     brkck           ; (exits concurrent if brk)
         bmi    exbrk           ;  exit if break...
        cmp     oup,x
        beq     wpu             ; wait for room (only happens if CONC mode)

        tay
        lda     tchar
        pha
wx49    sta     obuf,y          ; put character in buffer
        tya
        sta     ouend,x         ; save end ptr
         lda    #0
         sta    nochars,x       ; signal some chars in buffer
        pla

        bit     conflg          ; if not in conc mode
        bpl     isin            ;  then check if to flush buffer...

        cmp     #13             ; if CR then flush it...
        beq     fluit
         iny                    ; now check if buffer is full
```

```
         tya
         cmp    oup,x           ; check if this filled it
         bne    ret5            ; not full, so return with good status
fluit     jmp   flush           ; flush buffer...

isin     jsr    enaoi           ; enable output irq if chars
ret5     ldy    #1              ;  and return with good status
exbrk    rts


;        Get character
;        -------------

rget     jsr    getun           ; get unit number from table
         cpy    conflg          ; make sure in concurrent mode
         bne    ret8            ; jump if error...

cbk      jsr    brkck           ; check if break
         bmi    ret8            ;  jump if break error
         ldx    inp             ;  check if character in buffer
         cpx    inend
         beq    cbk             ; jump if none

         lda    ibuf,x
         inx
         stx    inp             ; get char and bump ptr
         sta    tchar           ; save character

         lda    tramod,y        ; check parity
         and    #$0C
         beq    ctra            ; jump if no parity check..
          cmp   #$0C
          beq   clpar           ; clear parity
         and    #4
         eor    tchar           ; force into even/ODD parity
ckpl     bcc    noe
          eor   #$80
noe      asl    a
         bne    ckpl
          bcc   clpar

         lda    #$20            ; set parity error
         ora    errflg
         sta    errflg

clpar    asl    tchar
         lsr    tchar           ; clear parity

ctra     lda    tramod,y        ; get translation mode
         and    #$30
         tax
         lda    tchar
         cpx    #$20
         bcs    notr            ; jump if no translation
          and   #$7F
          cmp   #13             ; CR
          bne   nocr
           lda  #$9B
nocr     dex
         bmi    notr            ; exit if not heavy

         cmp    #' '            ; heavy translation
```

```
        bcc     gdef
        cmp     #$7D
        bcc     notr
gdef    lda     trachr,y        ; translation character
notr    ldy     #1
        rts

ret8    jmp     restor          ; take it out of concurrent


        sbttl   'Interrupt handlers'
        page

;       Check break flag.. if break then abort
;       --------------------------------------

brkck   bit     brkflg          ; check break flag
        bmi     reno
        jsr     restor          ; restore from concurrent
        ldy     #$80            ; reset break flag...
        sty     brkflg          ; S=1 if error (Y=error msg)
        rts
reno    bit     restor          ; force S flag to 0
        rts


;       serial output ready/complete IRQ
;       --------------------------------

sordy
socmp   CLD                     ; make sure decimal flag is cleared
        tya
        pha                     ; save Y register (but not A)
        txa
        pha

        ldy     conflg          ; get correct buffer...
        lda     hiad,y          ; get high address of buffer
        sta     WX79+2          ; save it in routine...
        lda     oup,y           ; see if is a character to send
        cmp     ouend,y
        bne     seou            ; jump if so.. send character

        lda     pokmsk          ; clear interrupt flag
        and     #$E7
        sta     pokmsk
        sta     irqen
         lda    #-1
         sta    nochars,y       ; no characters in buffer
         bne    noinr           ; jump (always)

seou    tax
        inx                     ; get character..
wx79    lda     obuf,x
        sta     seroutr
        txa
        sta     oup,y

wtod    lda     #$08            ; make sure an Output Done interrupt
        and     irqst           ;  will not occur.  (apparently OD can
        beq     wtod            ;  happen even after SEROUT is loaded)
```

```
noinr   pla
        tax

reti    pla                     ; return from interrupt
        tay
        pla
        rti


;       Serial input ready IRQ
;       ----------------------

sirdy   CLD                     ; make sure not in deciaml mode
        tya
        pha

        lda     serinr          ; get character
        ldy     inend           ; get input buffer ptr
        sta     ibuf,y

         lda    skstat          ; get status
         sta    skres           ;  ..reset status
          eor   #-1
          and   #$C0            ; get overrun/frame error bits
          ora   errflg          ; save flags

        iny
        cpy     inp             ; check if hit end..
        sty     inend
        bne     exin            ;  jump if no overrun..
         iny                    ; bump current position
         sty    inp
         ora    #$10            ; set overrun flag
exin    sta     errflg
        clc
        bcc     reti            ; return from interrupt


        sbttl   'Tables and system variables'
        page


;       Table for DCB on status command
;       -------------------------------

sttab   db      'S',$40
        dw      stloc,4,4

;       Baud rate tables
;       ----------------

bautl   db      $A0,$CD,$E3,$6F,$95,$C0,$F6,$47
        db      $A0,$CC,$E3,$EA,$6E,$B3,$56,$28
bauth   db      $0B,$4C,$45,$3D,$2E,$1F,$19,$17
        db      $0B,$05,$02,$01,$01,$00,$00,$00


xwarm   db      -1              ; first time through, no dosini is done

tramod  db      $00,$00         ; translation mode:
                                ;   B6 - 1 if append LF after CR (out)
                                ;   B5 - 1 if no translation
```

```
                                ;    B4- 1 if heavy, 0 if light
                                ;   B[3,2] - 00 ignore parity (input)
                                ;             01 check odd/ clear
                                ;             10 check even/ clear
                                ;             11 no check/ clear
                                ;   B[1,0] - 00 no change (output)
                                ;             01 set odd parity
                                ;             10 set even parity
                                ;             11 set parity bit to 1
baudr   db      $00,$00         ; Baud rate:
                                ;   B[3-0] - baud rate (index into table)
                                ;   B[5-4] - 00 = 8 bits
                                ;             01 = 7 bits
                                ;             10 = 6 bits
                                ;             11 = 5 bits
                                ;   B[7] - 1 if 2 stop bits (else 1 stop)
ctsflg  db      $00,$00         ; indicates which must be true for conc mode
                                ;   B[0] - 1 if CRX monitor
                                ;   B[1] - 1 if CTS monitor
                                ;   B[2] - 1 if DSR monitor
trachr  db      $00,$00         ; Translation character: char to return
                                ;  if in heavy translation..
ecode                           ; address of end of handler code

                        ; THE FOLLOWING MUST STAY IN ORDER...
                        ; (and after TRACHR)!!!
conflg  ds      1               ; Concurrent mode flag.. (<0 if not in conc)
inp     ds      1               ; input ptr
inend   ds      1               ; input end ptr
oup     ds      2               ; output ptr
ouend   ds      2               ; output end ptr
nochars ds      2               ; no chars in out buffer flag

iodir   ds      8               ; icax1z bytes (saved from open)
unit    ds      8               ; icdnoz bytes (saved from open/xio/status)
tpok    ds      6               ; holds serial I/O irq vectors
errflg  ds      1               ; error flag

obuf    ds      256             ; output buffer..
        ds      256             ; 2nd output buffer
ibuf    ds      256             ; input buffer..
cend                    ; End of handler.. (new memlo)

        end
```

APPENDIX D—STANDARD PRINTER & MODEM CABLES

The following is the standard connection specification used by ICD for our standard printer and MODEM cables. These should work for the most common printers and MODEMs or they may need to be modified according to the special needs of your particular installation

Printer Cable Connections

36 pin centronics (male)	DB15P
1	1 - Data Strobe
2	2 - D0
3	3 - D1
4	4 - D2
5	5 - D3
6	6 - D4
7	7 - D5
8	8 - D6
16	11 - Gnd
32	12 - Fault
11	13 - Busy
9	15 - D7
Frame - to the shield wire	No connection to shield

MODEM Cable Connections

DB25P	DB9P
20	1 - DTR
8	2 - CRX
2	3 - XMT
3	4 - RCV
7	5 - GND
6	6 - DSR
4	7 - RTS
5	8 - CTS
Frame - to the shield wire	No connection to shield

APPENDIX E—1200XL MODIFICATIONS

WARNING: The following instructions should help anyone competent with soldering equipment to modify the 1200XL to work with the P:R: Connection and other computer powered peripherals. This modification is not intended for the complete novice.

Turn your computer on its back. **Remove the six phillips head screws which hold the case together** and place them in your parts dish. Turn the computer right side up and lift the top cover up and towards the front. Look inside and find the two ribbon cables which connect the keyboard and console LEDs to the main computer board. Carefully unplug these cables noting the correct polarity of their connectors. **Remove the keyboard assembly and set it aside for now.**

Remove the six phillips head screws holding the computer board in the bottom case. One of these screws is in the upper left hand corner near the on/off switch. Another is in the upper right corner and goes through the heat sink. The remaining four screws are across the front and about four inches apart. (Three of these also hold down the metal shield.)

Remove the computer board assembly from the case. Lift the front of the computer board and the cartridge/joystick/switch assembly up and pull the computer board out and towards you until all the rear connectors are free. Remove this assembly, separate the plastic piece from the PCB and set it aside.

Remove the metal shields and set them aside. There should be several "push" rivets. Remove these then separate and remove the metal covers. NOTE: Some metal covers may be held together with bent metal tabs or screws.

Replace resistor R63 with a jumper wire. 1200XL is the only 8-bit Atari computer with a current limit resistor (R63). This prevents 1200XL owners from using any peripherals (including the XM301 MODEM and P:R: Connection) which draw power from the computer. R63 is located at the top of the PCB near the center. It is just to the right of transistor Q3. Remove this resistor and replace it with a jumper wire. (Any piece of 24-30 gauge wire will do.)

Reassemble and test.
Now you can use devices which draw power from your 1200XL!

APPENDIX F—COMPATIBILITY

The following is a list of MODEM software which has been found to be incompatible with the P:R: Connection handler and our fix or solution to the problem.

TSCOPE by Joe Miller

TSCOPE was designed several years ago as a terminal program to support COMPUSERVE Vidtex (a trademark of COMPUSERVE) file transfer protocol. It only works with the 850 (not the Atari direct connect MODEMs) and is used by many COMPUSERVE subscribers. Joe Miller has graciously modified TSCOPE into RSCOPE which will now support the P:R: Connection as well as the 850. RSCOPE is in the public domain and available for downloading from various boards. A copy of RSCOPE is included on our P:R: Connection distribution diskette. Operation is the same as for TSCOPE.

The Learning Phone by Atari

The Learning Phone is an obsolete cartridge program written by the old Atari as the terminal program to support PLATO. Two years later it was finally released by the new Atari. Unfortunately, the learning phone does not allow use of either the P:R: Connection or SmartMODEMs (the type of MODEM which most interface users will own). John Skruch of Atari Corp. has assured us that a new Learning Phone will support the P:R: Connection along with SmartMODEMs.

Hometerm by Russ Wetmore

Hometerm published as part of Homepack by Batteries Included is one of the more popular MODEM programs. It uses an internal 'RBIN' type handler and does not load or use the internal 850 handler. PRC.SYS was written to provide 850 SIO emulation for this program (as well as others which may use the internal 'RBIN' approach). To use PRC.SYS you can either load it first as part of a batch file or append your terminal program to the end of it (in the case of an AUTORUN.SYS).

IMPORTANT WARRANTY INFORMATION
LIMITED 30 DAY WARRANTY

ICD, INC. warrants to the original consumer purchaser that this **ICD, Inc.** Personal Computer Product (not including computer programs) shall be free from any defects in material or workmanship for a period of 30 days from the date of purchase. If any such defect is discovered within the warranty period, **ICD, Inc.'s** sole obligation will be to repair or replace, at its election, the Computer Product free of charge on receipt of the unit (charges prepaid, if mailed or shipped) with proof of date of purchase satisfactory to **ICD, Inc.**

Write to:

ICD, Inc.
1220 Rock Street, Suite 310
Rockford, IL 61101-1437
Attn: Service Dept.

YOU MUST RETURN DEFECTIVE COMPUTER PRODUCT FOR IN-WARRANTY REPAIR.

This warranty shall not apply if the Computer Product: (i) has been misused or shows signs of excessive wear, (ii) has been damaged by improper installation, or (iii) has been damaged by being serviced or modified.

ANY APPLICABLE IMPLIED WARRANTIES, INCLUDING WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE HEREBY LIMITED TO THIRTY DAYS FROM THE DATE OF PURCHASE. CONSEQUENTIAL OR INCIDENTAL DAMAGES RESULTING FROM A BREACH OF ANY APPLICABLE EXPRESS OR IMPLIED WARRANTIES ARE HEREBY EXCLUDED. Some states do not allow limitations on how long an implied warranty lasts or do not allow the exclusion or limitation of incidental or consequential damages, so the above limitations or exclusions may not apply to you.

This warranty gives you specific legal rights and you may also have other rights which vary from state to state.

DISCLAIMER OF WARRANTY ON ICD, INC. COMPUTER PROGRAMS: All **ICD, INC.** computer programs are distributed on an "as is" basis without warranty of any kind. The entire risk as to the quality and performance of such programs is with the purchase. Should the programs prove defective following their purchase, the purchaser and not the manufacturer, distributor, or retailer assumes the entire cost of all necessary servicing or repair.

ICD, Inc. shall have no liability or responsibility to a purchaser, customer, or any other person or entity with respect to any liability, loss, or damage caused directly or indirectly by computer programs sold by **ICD, Inc.** This disclaimer includes but is not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of such computer programs.

REPAIR SERVICE: If your **ICD, Inc.** Personal Computer Product requires repair other than under warranty, please write to **ICD, Inc.**, Service Department for repair information.

IMPORTANT: If you ship your **ICD, Inc.** Personal Computer Product, package it securely and ship it, charges prepaid and insured, by parcel post or United Parcel Service.

WARRANTY/UPDATE REGISTRATION CARD

Please take the time to complete this card and return it to us to allow us to provide you with more efficient service, including updates, should your **ICD, Inc.** product require it.

(Please print)

Name_____

Address_____

City_____ State_____

Country_____ ZIP_____

Phone ( ) _____ Item Purchased
(Area Code)

Date of Purchase_____ Serial Number_____

Where Purchased_____

What other products would you like to see us develop?

_____

_____

_____

Does your local Atari dealer carry our product line? | | Yes | | No

Your Atari dealer's name, address_____

_____

_____

_____

PLACE
STAMP
HERE

ICD, Inc.
1220 Rock Street, Suite 310
Rockford, Illinois 61101-1437

Possible Errors Using the P:R: Connection

CODE #	ERROR CODE MEANING
128($80)	Break Key Was Pressed
129($81)	IOCB Already Open
130($82)	Nonexistent Device
131($83)	Open for Write Only
132($84)	Invalid XIO Call Made
133($85)	IOCB Not Open (from CIO)
135($87)	Open for Read Only
138($8A)	Device Timeout
139($8B)	NAK - Input Handshake Lines Not Ready
153($99)	Already in Concurrent Mode

ICD, Inc., 1220 Rock Street, Rockford, IL 61101-1437 815/968-2228